Problem Statement and Goals
RoCam

Team #3, SpaceY
Zifan Si
Jianqing Liu
Mike Chen
Xiaotian Lou

Table 1: Revision History

Date	**Developer(s)**	**Change**
Sept. 8, 2025	Jianqing Liu	Initial Draft
Sept. 9, 2025	Zifan Si	Details Elaboration
Sept. 10, 2025	Zifan Si	Fix based on teammate suggestion
Sept. 10, 2025	Jianqing Liu	Refine inputs, outputs, and goals
Sept. 12, 2025	Jianqing Liu	Update Depolyment Environment
Sept. 13, 2025	Xiaotian Lou	Update Citation
Oct. 4, 2025	Zifan Si	Fix based on TA feedback

1 Problem Statement

The problem statement outlines the motivation and context behind our project, RoCam (High Performance Vision-Guided Rocket Tracker). It defines the specific engineering problem of achieving reliable, real-time visual tracking for small-scale model rockets using a gimbal-mounted optical camera system. It also identifies the inputs and outputs of RoCam, such as live camera feeds, gimbal state data, and stabilized video streams, and states the performance objectives and key stakeholders involved in its development. The goal is to enable accurate mid-flight observation for analysis of staging and parachute deployment, thereby improving both safety and performance in future rocket launches.

1.1 Problem

In model rocketry, two of the hardest engineering problems are staging failures and parachute tangling. These problems are hard to study because they happen in mid-flight, where direct observation is limited. Without reliable tracking, engineers cannot fully understand these failures or design better solutions. As a result, recurring issues remain unresolved, slowing progress in both safety and performance of future rockets.

Model rockets travel at very high speeds, sometimes faster than Mach 3 and over 100 km in altitude (Space Concordia, 2025). Under these conditions, manual camera tracking is not possible.

Tracking a small model rocket is even harder than tracking large rockets such as the Falcon 9. Smaller size and uncontrolled launch conditions make accurate detection and continuous tracking much more difficult.

Some commercial tracking systems exist (AVe, 2021), but they lack the speed and precision needed for small, fast-moving rockets. To address this gap, our project, RoCam, aims to develop a real-time camera tracking system that can automatically follow model rockets during flight for clear mid-air observation.

1.2 Inputs and Outputs

- **Inputs**
  - Realtime camera feed of the rocket during flight
  - State information from the camera gimbal (a motorized mount that stabilizes and aims the camera)
  - Manual camera gimbal adjustment commands from user
- **Outputs**
  - Commands that move the camera gimbal to keep the rocket in frame
  - Real time video preview for the operator
  - Real time stabilized video output to the live streaming equipment

1.3 Performance Objectives

- The system should be able to process a 1080p 60 fps input video feed from the camera
- The System should be able to output 1080p 60fps to the live streaming equipment
- Real time video preview should be at least 15fps

1.4 Stakeholders

Direct Stakeholders

1. **McMaster Rocketry Team**: The main users of the system. They bring the system to the launch site, set up the tripod and gimbal, connect power, and check that the camera and encoders work. They open the UI, choose manual or auto tracking, and start or stop recording. During the flight, they watch the live preview and status, and they can ask the system to reacquire the rocket if it is lost. After the flight, they save the video and logs and use them to review staging, parachute events, and overall flight performance. They also perform basic maintenance, such as charging batteries, tightening mounts, and updating the software when needed.

2. **Faculty Supervisor (direct, limited use)**: The supervisor is part of the McMaster Rocketry Team and will use the system in a few clear ways that support safety and quality, below are how they will specifically interact with the system:

    - Before launch: Open the dashboard and read the simple pre-flight checklist. Confirm sensors show healthy, encoders are zeroed, storage has enough space, time is synchronized, and batteries are above the required level. Give an "OK to arm" in the UI so the team can proceed.
    - During the launch window: Keep the dashboard open to monitor live status, tracking lock, reacquisition attempts, dropped frames, and any warnings. If there is a safety concern or the system behaves poorly, press Hold or Abort and tell the team to pause.
    - After flight: Use the export button to save the video and telemetry logs to the shared folder. Mark key moments such as boost, apogee, parachute deployment, and landing. Add short notes about any issues seen during the run so the team can fix them.
    - Maintenance and follow-up: Review error logs and calibration results after field tests. Approve simple fixes, request a short re-test if needed, and sign off that the system is ready for the next launch.

Indirect Stakeholders

1. **Aerospace Engineers and Researchers**: Use the recorded video and telemetry to validate models, tune simulations, and improve rocket and recovery designs. They may request specific camera angles, overlays, or data formats to match their analysis tools.
2. **Event Organizers and Safety Officers**: Rely on clear tracking and status to confirm safe flight and recovery. They may ask the team to show live status, prove that the system is armed only when allowed, and keep an audit of warnings and holds.
3. **Engineering and Robotics Community**: Reuse design ideas such as the two-axis gimbal, the simple operator UI, the acceptance tests, and the logging format. They may adapt the approach for UAV tracking, sports analysis, or robot vision demos.
4. **Potential Commercial and Industrial Users**: Apply the system to tasks that need real-time, vision-guided tracking, such as site inspection, wildlife monitoring, or basic perimeter watch. They may care about reliability, battery life, and clear procedures for setup and export.

1.5 Deployment Environment

- Outdoor rocket launch sites with variable lighting, wind, and dynamic backgrounds.
- Indoor lab-based testing and demonstration for development and evaluation.

2 Goals

The goal of this project is to develop a software+hardware stack that is able to command a gimbal with a camera mounted on it to track and record model rocket launches.

This system aims to provide insights into the flight performance by having high quality flight footage.

The goal of this project is to design and implement RoCam, a real-time camera tracking system that automatically detects and follows small-scale model rockets during launch, ascent, and descent (apogee $<$ 200 m). The system aims to maintain a stable visual lock on the rocket using a motorized gimbal, process a 1080p 60 fps video feed in real time, and output both live and recorded stabilized footage. By meeting these goals, RoCam will provide clear mid-flight visual data to support analysis of staging events, parachute deployment, and overall flight performance.

3 Stretch Goals

- Handle 4K 60fps camera stream.
- Include functionalities to control the zoom and the focus of more advanced cameras.
- Integration with a full-size gimbal developed by the McMaster Rocketry Team.
- Track high-powered model rocket launches (apogee 3km+), while achieving all the performance objectives listed in Section 1.3.

4 Extras

In addition to the core goals of RoCam, our team plans to include a few value-added components that enhance the overall quality and usability of the system. These extras aim to demonstrate practical integration between hardware and software, improve accessibility for end users, and provide clearer insight into the system's design and operation. The selected extras include the design of the control circuit used to drive the gimbal actuators and the creation of a user instructional video for system setup and operation.

4.1 Circuit Design

As this project requires the use of a mechanical gimbal, an electrical circuit is needed to interface between the host computer and the gimbal actuators. The circuit will handle motor control, feedback sensing, and communication with the main software to enable precise and stable camera movement.

4.2 User Instructional Video

An user instructional video will be created to help potential users to understand how to use the system.

Appendix — Reflection

Team Reflection (Q1–Q3)

1. What went well while writing this deliverable?
2. What pain points did you experience during this deliverable, and how did you resolve them?
3. How did you and your team adjust the scope of your goals to ensure they are suitable for a Capstone project (not overly ambitious but also of appropriate complexity for a senior design project)?

Team Summary Our team collaborated effectively on this deliverable, maintaining clear communication and giving constructive feedback to improve each section. The writing process went smoothly overall, and adding the "Performance Objectives" section helped clarify what we meant by "high performance" and made our project goals more specific. Some challenges arose, such as defining what "high performance" should mean for our system and deciding how complex the implementation should be, but we discussed these issues together and reached a clear, realistic definition supported by measurable goals. Learning to write in LaTeX and follow the template format also took time, but sharing examples and helping one another fixed most formatting issues quickly. As the main editor, keeping everyone's updates synchronized before the deadline was difficult, so we set small internal deadlines and used GitHub to track edits efficiently. Overall, the team worked cohesively, divided tasks fairly, and refined the document through supervisor and TA feedback to produce a clear and well-structured document.

Individual Reflections

Jianqing Liu from hardware

1. **Q1 (What went well?):** I drafted most sections and kept edits moving fast by sharing early versions and asking for quick comments. Section 1.1 (*Problem*) improved a lot after I rewrote it with simpler words and a short example of why manual tracking fails. Adding the *Performance Objectives* made our goals clearer to non-rocket readers. The team gave focused feedback, and I merged changes the same day to keep one clean version. Using Git branches for small edits avoided merge conflicts.
2. **Q2 (Pain points & resolution):** Defining "high performance" was hard because it could mean many things. We listed what the operator actually needs in the field (smooth video, low delay, stable tracking) and turned that into numbers (60 fps, $<$120 ms latency, $\leq 1.5^\circ$ error). I also struggled with LaTeX formatting at first; setting up a simple compile script and a small style guide fixed most issues. When we disagreed on the environment constraints, we asked a TA to confirm what is realistic for a capstone. After that, writing went smoother.
3. **Q3 (Scope adjustment):** I pushed to keep the core hardware simple: a two-axis gimbal with encoders and a stable mount. We moved custom PCB work and advanced sensors to "Extras" so our main goal stays on time. We agreed to prove value with one camera, one tracker, and clean video first. That plan still shows strong engineering work but avoids risky detours. If we finish early, we can add the extra hardware later.

Xiaotian Lou from QA

1. **Q1 (What went well?):** Clear task ownership and short check-ins helped us finish drafts without last-minute rush. I focused on quality: making sure terms were consistent and each claim matched the numbers in the objectives. We kept a simple checklist for each section (goal, assumptions, metric, test), which reduced back-and-forth. Getting a faculty advisor early also helped us avoid unrealistic promises. Overall, our writing became more direct and easy to read.
2. **Q2 (Pain points & resolution):** LaTeX slowed us down at first, especially tables and references. I set up a minimal template, editor plugins, and a compile command so anyone could build the PDF the same way. Some sections were vague (e.g., "robustness"), so I ran short working sessions to turn

them into testable statements (temperature, wind, glare). I also added a quick pass for grammar and formatting to keep the tone consistent. These steps made the document feel unified.

3. **Q3 (Scope adjustment):** From a QA view, success must be testable. I proposed acceptance tests tied to each metric: lock percentage during ascent, glass-to-glass latency, and reacquire time. We added operating bounds to avoid undefined cases in testing (e.g., temperature range, wind limit). I recommended deferring multi-target tracking and night IR, since they would add many new tests. This keeps the test plan realistic and still rigorous.

Shike Chen from CV

1. **Q1 (What went well?):** Team alignment made the outline simple, which helped me describe the vision pipeline clearly. We agreed to start with a reliable detector and clean data path before trying advanced models. I wrote short explanations of bbox, centroid, and control loop so the reader can follow without CV background. The link between CV output and gimbal command is now easy to see. Early diagrams and plain words reduced later rewrites.

2. **Q2 (Pain points & resolution):** "High performance" meant different things to us at first. I translated CV needs into numbers: minimum fps, acceptable delay from frame to command, and how much jitter is okay. We added a small telemetry log requirement so we can verify results after a test. I also simplified the wording in the CV section to avoid heavy jargon. That made reviews faster and less confusing.

3. **Q3 (Scope adjustment):** To keep work realistic, I suggested we focus on a single-rocket day-time tracker with good reacquire, instead of multi-target or long-range RF links. We fixed baseline targets (60 fps, $<$120 ms latency, $\leq 1.5°$ error) and kept model choice flexible as long as it meets those numbers. Hard features like night scenes or strong smoke are now future work. This plan lets us deliver a strong demo and solid data first.

Zifan Si from software

1. **Q1 (What went well?):** Separating the "why" (motivation) from the "how" (design) made the writing clearer. I tied software pieces to user needs: live preview, start/stop, reacquire, and clear errors. Linking each feature to a metric or acceptance test kept the document consistent. I also helped set up a simple folder structure and a build script so teammates could update the PDF easily. Small, frequent commits kept everyone in sync.

2. **Q2 (Pain points & resolution):** Balancing ambition with field limits was hard. I proposed explicit operating bounds (temperature, wind, sun angle) and a reacquire target so the scope is realistic. We wrote down latency sources (capture, processing, encode, UI) to keep the end-to-end delay under control. LaTeX details took time, but shared examples and a short style guide fixed most issues. After that, edits were faster and more consistent.

3. **Q3 (Scope adjustment):** I recommended keeping the core deliverable to autonomous tracking plus stabilized recording with a simple operator UI. More advanced items—custom control PCB, complex RF, or multi-target—were moved to Extras. We also agreed to log key timings and states so we can prove performance. This plan keeps risk low while still leaving room for stretch goals if time allows.

References

AI Auto Tracking PTZ Camera TR3XX Series User Manual. AVer Information Inc., October 2021. Version 1 (2021-10-15).

Space Concordia. Space concordia rocketry. https://web.archive.org/web/20250717115148/https://spaceconcordia.ca/rocketry, 2025. Archived 2025-07-17; accessed 2025-09-13.